Modeling of Secure and Dependable Applications Based on a Repository of Patterns: The SEMCO Approach

Brahim Hamid
IRIT- University of Toulouse
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
hamid@irit.fr

***Abstract*—The requirement for higher quality and seamless development of systems is continuously increasing, even in domains traditionally not deeply involved in such issues. Security and Dependability (S&D) requirements are incorporated to an increasing number of systems. These newer restrictions make the development of those systems more complicated than conventional systems. In our work, we promote a new approach called SEMCO (System and software Engineering with Multi-COncerns) combining Model-Driven Engineering (MDE) with a model-based repository of S&D patterns to support the design and the analysis of pattern-based secure and dependable system and software architectures.**

The modeling framework to support the approach is based on a set of modeling languages, to specify security and dependability patterns, resources and a set of property models, and a set of model transformation rules to specify some of the analysis activities. As part of the assistance for the development of S&D applications, we have implemented a tool-chain based on the Eclipse platform to support the different activities around the repository, including the analysis activities. The proposed approach was evaluated through a case study from the railway domain.

Keywords*— *Security, Dependability, Resource, Pattern, Model-driven Engineering, Embedded Systems Engineering.

I. Introduction

During the last decades, the systems have grown with an increasing in terms of complexity and connectivity. In the past Security and Dependability (S&D) was not such a critical concern of system development teams, since it was possible to rely on the fact that a system could be easily, controlled due to its limited connectivity and, in most of the cases, its dedicated focus. However, nowadays, systems are growing in terms of complexity, functionality and connectivity not only in safety-critical areas (defense, nuclear power generation, etc.), but also in areas such as finance, transportation, medical information management and system using web applications.

Just consider Resource Constrained Embedded Systems (RCES) [1] and their added complexity and connectivity. The aforementioned challenges in modern system development push the Information and Communication Technologies (ICT) community to search for innovative methods and tools for serving these new needs and objectives. Regarding system security and dependability, in the cases of modern systems, the "walled-garden" paradigm is unsuitable and the traditional security and dependability concepts are ineffective, since it was based on the fact that it is possible to build a wall between the system and the outer world. In our opinion the foundation for comprehensive security and dependability engineering is a deep understanding of the modern systems, ongoing or previous security and dependability incidents and their implications on the underlying critical infrastructure.

The industrial context conducting our work is how to take into account several constraints, mainly those related to security and dependability, that are not satisfied by the well-known and the widely used technology for building applications for Resource-Constrained Embedded Systems. These requirements introduce conflicts on the three main factors that determine the cost of ownership of applications: (a) cost of the production, (b) cost of engineering and (c) cost of maintenance. In other words, systems with high dependability requirements for which the security level must be demonstrated and certified use almost exclusively technical solutions strongly oriented by the application domains. Applications based on these solutions are by definition dedicated, hardly portable between different execution platforms and require specific engineering processes. These specificities greatly increase the cost of the development in the different phases of their lifecycle.

Embedded systems share a large number of common characteristics, including real-time and physical constraints (e.g. temperature), as well as energy efficiency requirements. Specifically, Resource Constrained Embedded Systems refer to systems which have memory and/or computational processing power constraints computing resources of RCES, e.g. memory, tasks, and buffers, are generally statically determined. The generation of RCES therefore involves specific software building processes. These processes are often error-prone because they are not fully automated, even if some level of automatic code generation or even model driven engineering support is applied. Furthermore, many RCES also have assurance requirements, ranging from very strong levels involving certification (e.g. DO178 and IEC-61508 for safety-relevant embedded systems development) to lighter levels based on industry practices. Consequently, the conception and design of RCES is an inherently complex endeavor. To cope with the growing complexity of embedded system design, several development approaches have been proposed. The

most popular are those using models as main artifacts to be constructed and maintained.

In embedded system design field, the integration of non-functional requirements from Security and Dependability (S&D) [2], [3] are exacerbating this complexity, mainly in the context of trade-offs. For instance, in the automotive domain, a car may need to have secure communication mechanisms for secure download or for secure data transfer. In the railway domain, a supervision system needs to have secure communication mechanisms to be able to activate the emergency brake when something goes wrong. The development of systems with security and dependability requires specialized expertise and skills. The cost of designing such features from scratch could easily exceed the cost of the rest of the system. For example (see Fig. 1), the development of a security component for a railway signaling system requires expertise in security that is seldom available in the railway industry and it requires expertise in the engineering process and validation practices of railway which are not always available in the S&D community. In fact, capturing and providing this expertise by the way of security and dependability patterns can support the integration of S&D features by design to foster reuse during the process of software system development. Patterns are specified and validated by security and dependability experts and stored in a repository to be reused as security and dependability building block function by software engineer in several domains.

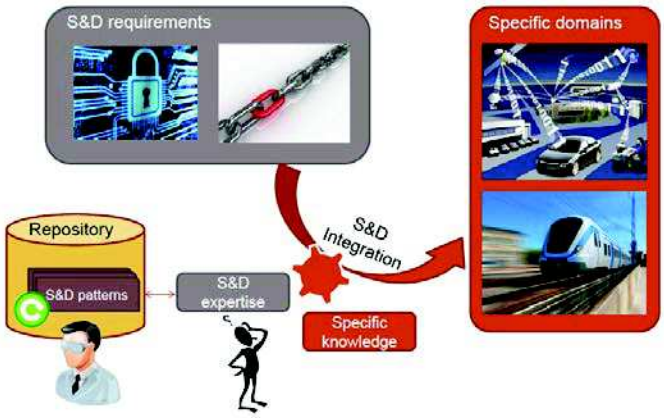

Fig. 1. Patterns for engineering systems with security and dependability requirements

Recent times have seen a paradigm shift in terms of design by combining multiple software engineering paradigms, namely, Model-Driven Engineering (MDE) [4] and Component Based Software Engineering (CBSE) [5]. Such a paradigm shift is changing the way systems are developed nowadays, reducing development time significantly.

In our work, we promote a new discipline for systems engineering around a model-based repository of modeling artifacts using a pattern as its first class citizen: *Pattern-based System Engineering (PBSE)*. The proposed approach called *SEMCO* for System and software Engineering with Multi-COncern [1] addresses two kinds of processes: the process of modeling artifacts development and system development with modeling artifacts. Therefore, we add a repository as a tier which acts as intermediate agent between these two processes.

[1] http://www.semcomdt.org/

A repository should provide a modeling container to support modeling artifacts lifecycle associated with different methodologies. The patterns that are at the heart of our system engineering process reflect design solutions at domain independent and specific level, respectively.

II. BACKGROUND

A. Incorporating Security and Dependability in System Engineering

In system engineering, security and dependability may be compromised in several system layers. Usually, security is considered when design decisions are made leading to potential conflicting situations. The integration of security and dependability features requires the availability of system architecture expertise, application domain specific knowledge and security expertise at the same time to manage the potential consequences of design decisions on the security of a system and on the rest of the architecture. For instance, at architectural level, incorporating security means to have a mechanism (it may be a component or integrated into a component). Development processes for system and software construction are common knowledge and mainstream practice in most development organizations. Unfortunately, these processes offer little support in order to meet security and dependability requirements. Over the years, research efforts have been invested in methodologies and techniques for secure and dependable software engineering, yet dedicated processes have been proposed only recently, namely OWASP's CLAS[2], Microsoft's SDL[3] and McGraw's Touchpoints[4].

In *SEMCO*, our aim is (1) to identify the commonalities, discuss the specificity of each approach, and (2) to evaluate the integration of the *SEMCO* outcomes in these process models. The overall goal in *SEMCO* is to support any security and dependability engineering process.

B. Pattern-Based Development

Patterns are widely used today to provide architects and designers with reusable design knowledge. They refer to triples that describe solutions for commonly occur- ring problems in specific contexts. There are patterns for generic architecture problems [6], for security [7] and for other non-functional requirements.

Pattern-based development has gained more attention recently in software engineering by addressing new challenges that are not targeted in the past. In fact, they are applied in modern software architecture for distributed systems including middlewares [8], and real-time embedded systems [9], and recently in security and dependability engineering [7]. The related approaches promote the use of patterns in the form of reusable design artifacts.

The supporting research activities in PBSE examine three distinct challenges: (a) mining (discovering patterns from existing systems), (b) hatching (selection of the appropriate pattern); (c) application (effective use during the system development process). These three challenges often involve

[2] http://www.owasp.org
[3] http://msdn2.microsoft.com/en-us/library/ms995349.aspx
[4] http://www.swsec.com/resources/touchpoints/

widely diverse core expertise including formal logic, mathematics, stochastic modeling, graph theory, hardware design and software engineering.

In our work, we study only the two last challenges, targeting the (i) development of an extendible design language for modeling patterns for secure and dependable distributed embedded systems [10] and (ii) a methodology to improve existing development processes using patterns [11]. The language has to capture the core elements of the pattern to help its (a) precise specification, (b) appropriate selection and (c) seamless integration and usage. The first aspect is pattern-definition oriented while the second and the third aspects are more problem-definition oriented.

Usually, these design artifacts are provided as a library of models (sub-systems) and as a system of patterns (framework) in the more elaborated approaches. However, there are still lacks of modeling languages and/or formalisms dedicated to specify these design artifacts and the way how to reuse them in software development automation. More precisely, a gap between the development of systems using patterns and the pattern information still exists.

The *SEMCO* vision is to use S&D and architecture patterns and their interactions as parameters for the computation, analysis, selection and development of secure and dependable system and software architectures.

C. Model Driven Engineering (MDE)

MDE has the potential to greatly ease daily activities of S&D experts. In fact, MDE supports the designer to specify in a separate way S&D requirements issues at a greater abstraction level. MDE promotes models as first class elements. A model can be represented at different levels of abstraction and the MDE vision is based on (1) the metamodeling techniques to describe these models and (2) the mechanisms to specify the relations between them. Model exchange is within the heart of the MDE methodology as well as the transformation/refinement relation between two models. Domain Specific Modeling Languages (DSML) [12] in software engineering is used as a methodology using models as first class citizens to specify applications within a particular domain. There are several DSML environments, one of them being the open- source Eclipse Modeling Framework (EMF) [13]. EMF provides an implementation of EMOF (Essential MOF), a subset of the Meta Object Facility (MOF)[5], called Ecore. EMF offers a set of tools to specify metamodels in Ecore and to generate other representations of them. Query View Transformation (QVT)[6] is a standard to specify model transformations in a formal way, between metamodels conforming to MOF.

In the context of *SEMCO*, design decisions are one of the most important artefacts during architecting. Models of both security and dependability decisions and other architecture concerns decisions need to be complete, and need to be specified precisely and traced to other models. The *SEMCO* vision is that metamodeling and model transformation, within MDE (specification, design, analysis, implementation, test), allows reducing time/cost of understanding and analyzing system artefacts description due to the abstraction mechanisms and reducing the cost of development process thanks to the generation mechanisms.

[5] http://www.omg.org/spec/MOF/
[6] http://www.omg.org/spec/QVT/

III. THE SEMCO APPROACH

A. Objectives

SEMCO (System and software Engineering with Multi-COncern) aims at developing a model and pattern-based modeling framework for handling security and dependability for system and software architecture that semi-automatically supports the analysis and evaluation of secure and dependable architectures for verification and validation purposes, providing the subsequent re-design that optimizes both.

The framework provides several artifacts types representing different engineering concerns (Security, Dependability, Safety and Resources) and architectural information. These artifacts are stored in a model-based repository and provided in the form of modeling languages (*SEMCO-ML*), tools (*SEMCO-MDT*) and methods (*SEMCO-PM*). The nearest goal of *SEMCO* is going to contribute on “Understanding System and software Engineering with security and dependability features by design in resource constrained systems”.

B. Our Approach Through an Example: The Stakeholders

We propose a solution based on the reuse of software subsystems that have been pre-engineered to adapt to a specific domain. In order to understand our security and dependability engineering framework with patterns better, we provide a description of a one usage scenario.

In the example of Fig. 2, first a security expert develops a security subsystem called pattern. The security expert focuses mainly on security solution development in the form of patterns elements or mining patterns from existing systems. Next a software engineering expert adapts the pattern to engineering reuse. The main output of this activity is a specification of a pattern in suitable format for repository storage, to enforce reuse during system and software development processes. The activity of creating the blue artifacts is performed by the software engineer in collaboration with the security expert. The achieved role can be called a security and dependability pattern engineer. Then a domain process expert, for instance a railway domain expert adapts the security pattern into a version that is usable in the railway system development process, ensuring compliance of these artifacts with appropriate standards and that other engineering artifacts are available throughout each phase of a development process, creating the red artifacts.

Moreover, with the help of a software engineering expert, the pattern and its associated artifacts should be transformed into a version (green artifacts) that is adapted to the railway development environment. The activity of reusing the red artifacts is performed by dedicated tools that are customized for a given software engineering environment (development platform). Finally, a domain engineer, for instance a railway system and software developer reuses the resulting adapted and transformed pattern (green artifacts) to develop a railway system.

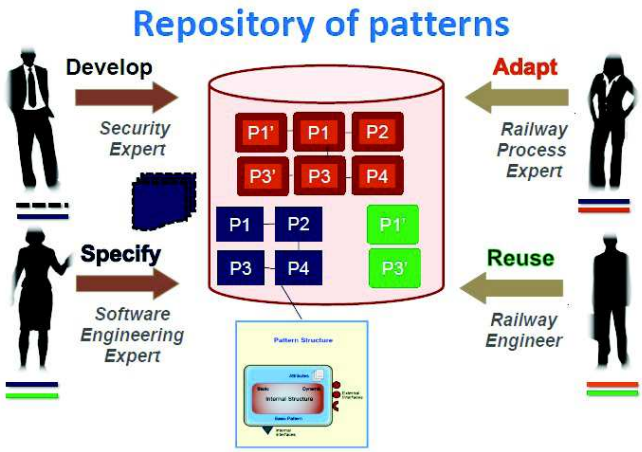

Fig.2. Our approach through an example: The Stakeholders

C. *Conceptual*

The *SEMCO* foundation is a model-based repository of modeling artifacts, including pattern, resource and property models and thereby supporting a pattern-based development methodology. The pattern is the first class citizen of these modeling artifacts to describe security and dependability solutions. The resource will capture the computing system platform and the property will allow to govern the use of patterns and to evaluate their security level for analysis for reuse. Specifically, we tend to overlook the three rules that govern pattern-based system development (1) the specification of these artifacts at different levels of abstraction, (2) the specification of relationships that govern their interactions and complementarity and the specification of the relationship between patterns and other artifacts manipulated during the development lifecycle and those related to the assessment of critical systems. It is a good application and promotion of model- driven engineering.

In *SEMCO*, a pattern is a subsystem dedicated to security and dependability aspects [14], to be specified by a security and dependability experts, and reused by domain engineers to improve systems/software engineering facing security and dependability requirements.

The core of *SEMCO* is a set of DSMLs, a repository, search and transformations engines. The DSMLs are devoted to specify patterns, a system of patterns and a set of models to govern their use, and thereby to organize, analyze, evaluate and finally validate the potential for reuse. In order to enforce reuse and to interconnect the process of the specification of these modeling artifacts and the system development with these artifacts, we developed a structured model-based repository to store these artifacts. Therefore, instead of defining new modeling artifacts, that usually are time and effort consuming as well as error prone, the system developer merely needs to select appropriate patterns from the repository and integrate them in the system under development. This is the role of search and transformation engines, where an artifact is identified/ selected from a repository and then the results are transformed towards specific domain development environments such as UML.

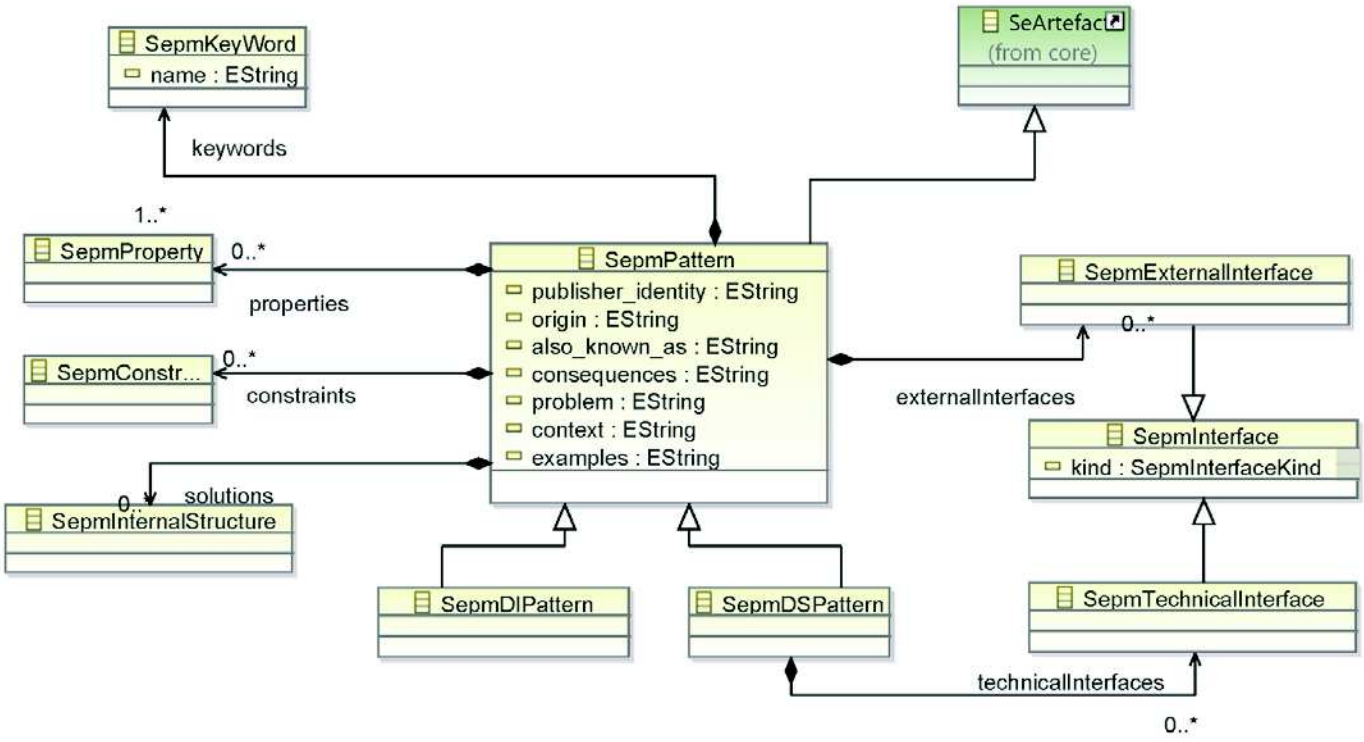

Fig. 3. The SEPM metamodel -Overview

IV. TECHNICAL FOUNDATION

The *SEMCO* vision is to create an integrated set of software tools to enable S&D RCES applications development by design, with the following objectives:

- The tools will improve the design, implementation, configuration and deployment of S&D RCES applications through:
  - Best-practice design methods: patterns and models
  - Innovative modeling and optimization techniques: Model Driven Engineering (MDE), Domain Specific Modeling Languages (DSML),
  - To foster reuse in multiple domains: repository.
- The tools will target multiple stakeholders in the RCES markets.
- The tools will provide and manage all interfaces with a common and evolving underlying core models and technologies.

As introduced in the previous section, security and dependability pattern engineer and the domain specific engineer use a number of tools. Those tools, based on model driven engineering techniques to create and then to reuse information that is stored in an engineering repository.

We now present an overview of our modeling framework building process as:

- *SEMCO-ML*: a set of DSMLs for the specification of the *SEMCO* modeling artifacts.
- *SEMCO-MDT*: a set of tools to support the *SEMCO* methods, the specification of the *SEMCO* modeling artifacts and the repository system.
- *SEMCO-PM*: a set of methodologies for the description of the PBSE methods.

Additional and detailed information will be provided during the implementation of the related design environment. Then, we detail the description of the integrated process used for the development of the Safe4Rail application in Section V.

A. SEMCO-ML

To foster reuse of patterns in the development of critical systems with S&D requirements, we are building on a metamodel for representing S&D patterns in the form of subsystems providing appropriate interfaces and targeting S&D properties. Interfaces will be used to exhibit the pattern's functionality in order to manage its application. In addition, interfaces support interactions with security primitives and protocols, such as encryption, and specialization for specific underlying software and/or hardware platforms, mainly during the deployment activity.

The *System and software Pattern Metamodel (SEPM)* is a metamodel defining a new formalism for describing patterns, while the *Generic Property Metamodel (GPRM)* is used to specify property model libraries to define the S&D and resource properties of the patterns. The principal classes of the SEPM metamodel are described with the Ecore notation in Fig. 3. In the following, we detail the meaning of principal concepts used to edit a pattern.

- *SepmPattern*. This block represents a modular part of a system representing a solution of a recurrent problem. A *SepmPattern* is defined by its behavior and by its provided and required interfaces. A *SepmPattern* may be manifested by one or more artifacts, and in turn, that artifact may be deployed to its execution environment. The *SepmPattern* has attributes to describe the related recurring design problem that arises in specific design contexts.

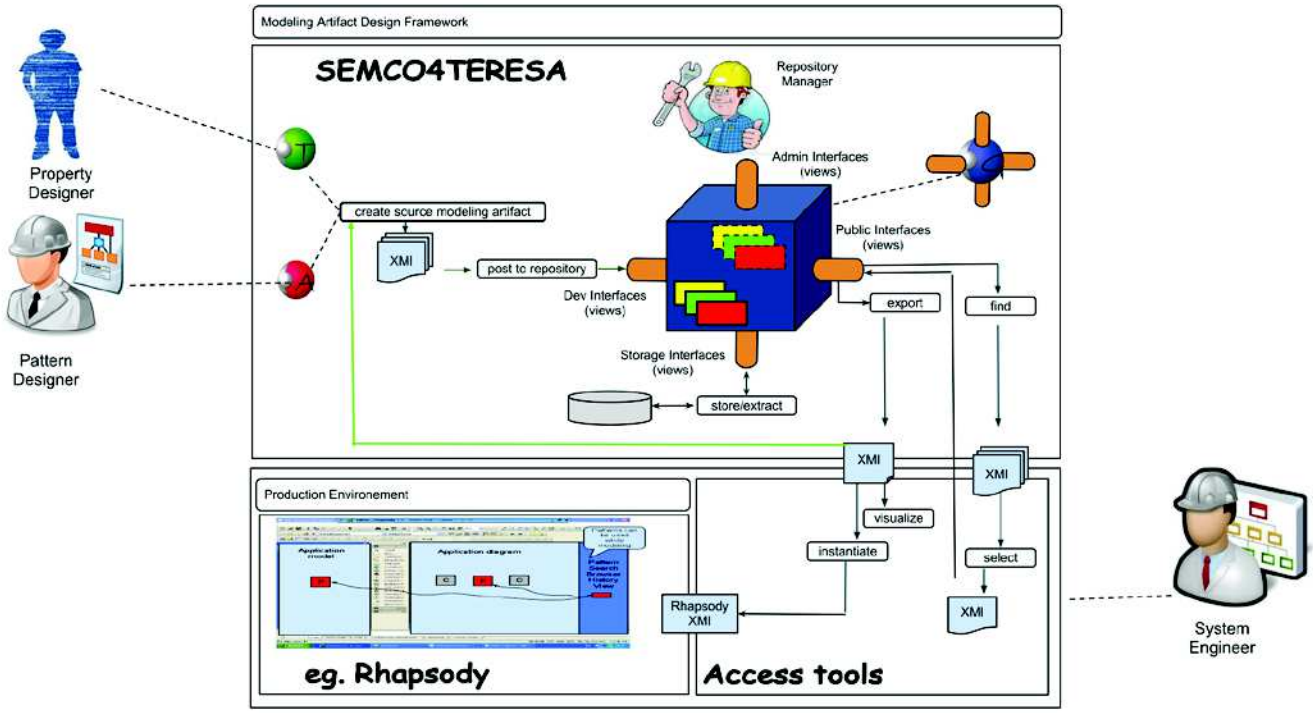

Fig. 4. The tool flow architecture

- *SepmInternalStructure.* Constitutes the implementation of the solution proposed by the pattern. Thus *the InternalStructure* can be considered as a white box which exposes the details of the pattern.
- *SepmInterface.* A pattern interacts with its environment with *Interfaces* which are composed of *Operations*. We consider two kinds of interface:
  (1) *SepmExternalInterface* for specifying interactions with regard to the integration of a pattern into an application model or to compose patterns, and
  (2) *SepmTechnicalInterface* for specifying interactions with the platform.
- *SepmProperty*. Is a particular characteristic of a pat- tern related to the concern dealing with and dedicated to capture its intent in a certain way. Each property of a pattern will be validated at the time of the pattern validation process and the assumptions used will be compiled as a set of constraints which will have to be satisfied by the domain application.

In addition to defining pattern artifacts, our pattern metamodel provides a way to formalize an *S&D Pattern System*, as a set of S&D patterns and their potential relationships, that enables an incremental support of our PBSE framework.

The *Generic Property (GPRM)* metamodel captures the common concepts of the two main concerns of trusted RCES applications: Security, Dependability and Resource on the one hand and Constraints on these properties on the other hand. The libraries of properties and constraints include units, types and categories. For instance, security and dependability attributes [5] such as authenticity, confidentiality and availability are categories of S&D properties. These categories require a set of measures types (degree, metrics…) and units (boolean, float…).

B. SEMCO-MDT

The tool-suite to support the *SEMCO* approach has to provide the following features:

- Repository life cycle: Allow the management of the repository, including deployment, set-up and organization.
- Modeling artefact life cycle: Provide the ability to editing S&D patterns and models, their validation, and their deposit in repository (DEP).
- System life cycle: Provide the ability to retrieve S&D patterns and models from repository (RET) by querying the repository, instantiate and integrate the results.

Using the proposed metamodels, the Eclipse Modeling Framework (EMF), and a CDO-based repository [7] ongoing experimental work is conducted with *semcomdt* (Semco Model Development Tools, IRIT's editor plugins) to produce an MDE Tool-chain, such as visualized in Fig. 4, supporting the approach. *semcomdt* provides a set of software tools, for instance for the design and for populating the repository and for retrieval and transform from the repository. For accessing the repository, *semcomdt* provides a set of facilities to help selecting appropriate patterns including *keyword* search, *lifecycle stage* search and property categories search. Currently the tool suite is provided as Eclipse plugins. For more details, the reader is referred to [15].

The following tools to perform the activities of management and populating the repository were developed:

- *Gaya:* a repository based on MDE technology was developed. This repository allows to store engineering and process knowledge associated with S&D patterns.
- *Arabion:* a tool for editing S&D patterns. These patterns must be stored in such a way that they can be reused later, enhanced and modified.
- *Tiqueo:* a tool for editing S&D properties and constraints. The focus is on the non-functional requirements that are associated with S&D patterns.

Moreover, a set of dedicated tools that are customized for a given software engineering environment (development platform) were developed: *Access tools* for Safe4Rail. The tool transforms the *Gaya* representation of S&D patterns into a representation that is consistent with the Safe4Rail set of tools (mostly Rhapsody UML-based) and the Safe4Rail process.

C. SEMCO-Methodology: From Pattern Repository to System Development

Along this description, we will give the main keys to understand why our process is based on a generic, incremental and a constructive approach. Once the repository[8] is available, it serves an underlying trust engineering process. In the process model visualized in Fig. 5 (the numbers in parentheses correspond to the numbers in Fig. 5), as activity diagram, the developer starts by system specification (A1) fulfilling the requirements. In a traditional approach (non pattern-based approach) the developer would continue with the architecture design, module design, implementation and test. In our vision, instead of following these phases and defining new modeling artifacts, that usually are time and effort consuming, as well as error prone, the system developer merely needs to select appropriate patterns from the repository and integrate them in the system under development.

For each phase, the system developer executes the search/select from the repository to *instantiate* patterns in its modeling environment (A4 and A9) and to *integrate* them in its models (A5 and A10) following an incremental process. The model specified in a certain activity is then used as an input work product in the following activities. Also, thanks to the system of patterns organization, the patterns identified in a certain stage will help during the selection activity of the following phases. Moreover, the system developer can develop their own solutions when the repository fails to deliver appropriate patterns at this stage. It is important to remark that the software designer does not necessarily need to use one of the artifacts stored in the repository previously included. He can define custom software architecture for some patterns (components), and avoid using the repository facilities (A6 and A11).

[7] http://www.eclipse.org/cdo/

[8] The repository system populated with S&D Patterns.

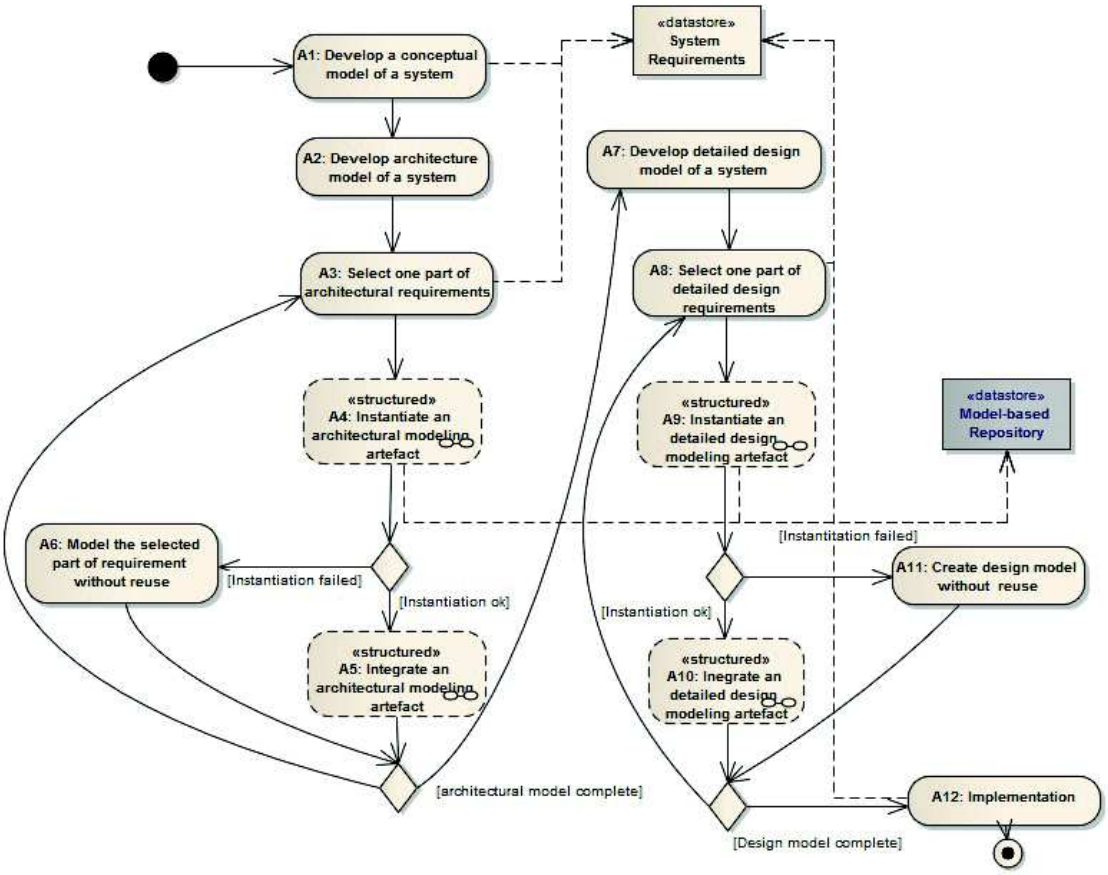

Fig. 5. The S&D pattern-based development process

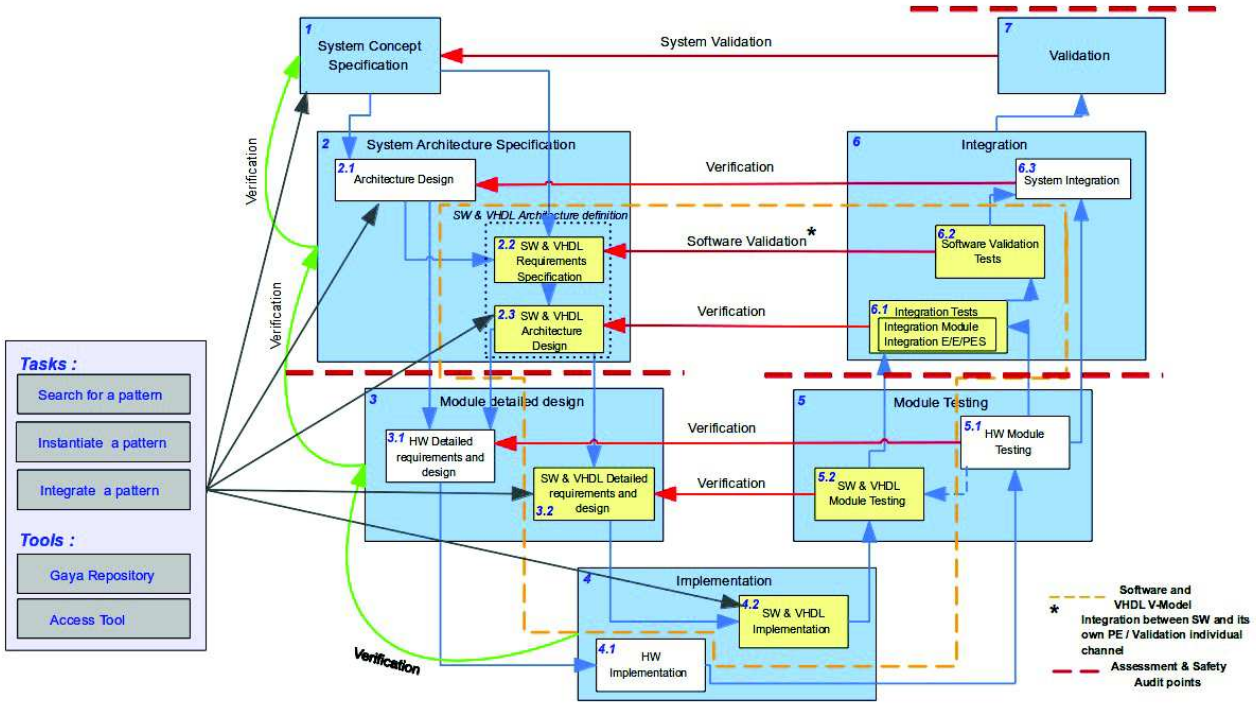

Fig. 6. An example of a railway engineering process

V. Application Domain Example

SEMCO is an effective approach, relying on an MDE tool-suite, to supporting secure and dependable system and software architecture engineering methodology and thus in our context supporting automated pattern-based building and access in industry[9]. We discuss the benefits, such as reduced modeling effort and improved readability, achieved when applying the methodology to an industrial case study. We have used the *SEMCO* modeling framework to model and to analyze secure and dependable pattern- based architectures for an application acting as one of the *TERESA* demonstrators for the railway domain called Safe4Rail. Safe4Rail is in charge of the emergency brake of a railway system. Its mission is to check whether the brake needs to be activated. Most important, the emergency brake must be activated when something goes wrong.

A. The TERESA Repository

An instance of the Gaya repository called *teresaRepository* was built in the context of the *TERESA* project to demonstrate reuse in a railway engineering environment and a metrology engineering environment. The railway and metrology domains analysis lead to identify a set of patterns to populate *teresaRepository*. We used the *Tiqueo* editor and *Arabion* editor to create the corresponding property libraries and the set of patterns, respectively. *Arabion* use the property libraries provided by *Tiqueo* to type the patterns property. Finally, we used the *Gaya* manager tool to set the relationships between the patterns.

The TERESA repository contains so far (on January 2014):

- *Compartments*. 21 compartments to store artifacts of the *TERESA* domains.
- *Users*. 10 users.
- *Property Libraries.* 69 property model libraries.
- *Pattern Libraries.* 59 patterns.

B. Application of the SEMCO Approach to a Railway System Case Study

Here, we examine the process flow for the example following the design process of Section IV-C and a very strict engineering process was followed, such as visualized in Fig.6, where specific activities were performed in order to achieve certification using the presented approach (the numbers in parentheses correspond to the numbers in Fig. 5). In this case, SIL4 level is targeted.

Once the requirements are properly captured and imported into the development environment, the process can be summarized with the following steps:

Activity A2: Develop architecture model of a system: (A3) The analysis of the requirements results in the needs of an architectural pattern for redundancy. Thus, activity (A4) is the instantiation of S&D patterns from the repository using the repository access tools. The running of the Retrieval tool using keywords *Redundancy* and *SIL4*, suggests to use a TMR pattern at architecture level. In addition, some diagnosis techniques imposed by the railway standard are suggested, thanks to the repository structure and the support of the system of patterns organization for the railway application domain. (A5) Finally, at architecture level, we will integrate the following patterns: (a) TMR (searched by the System Architect), (b) Diagnosis techniques (suggested by the tool) and (c) Sensor Diversity (searched by the System Architect).

Activity A7: Develop design model of a system: This activity involves the development of the design model of the system. (A8) The analysis of the requirements the architecture model and the identified architectural patterns will help during the instantiation activity of the design phase (A9). Based on the selected patterns, the repository may suggest related or complementary patterns. For instance, if the TMR has been integrated, the following patterns may be proposed for the design model iteration: (d) Data Agreement, (e) Voter, (f) Black Channel and (g) Clock Synchronization.

VI. Conclusion and Discussion

A Pattern Based System Engineering (PBSE) methodology based on a repository was specified. This engineering methodology fully takes into account the need for separation of roles by defining three distinct processes, the pattern modeling process, the repository specification process, and the pattern integration process. The implementation of a PBSE for S&D patterns is discussed in detail through a use case from railway domain. A set of languages were specified for the specification of S&D patterns, of S&D properties, of processes, and of the repository structure and content. By developing an effective model- and pattern- based engineering approach, *SEMCO* will contribute to the establishment of security and dependability as an engineering discipline in the area of embedded systems.

Our objective is to design frameworks to assist system and software developers in the domain of security and safety critical systems to capture, implement and document distributed system applications. We worked on the “theory of how”. We addressed four fundamental questions: (1) what is design solutions for the precise and valuable specifications of patterns and how can a pattern be specified hierarchically with all its facets? (2) what is a repository of patterns, and how can repository be built and used to instantiate its content? (3) what is PBSE and (4) how can pattern be integrated into a system under development? We also worked on a “practice of how” by providing the *SEMCO* tool-chain. We studied the first three questions, and we are now confronted with the fourth question.

We plan to extend this work in the following directions: We will investigate new design techniques to improve the pattern’s representation to ease their integration in existing software engineering processes targeting secure and dependable architectures. Furthermore, we will study the relation of our approach to the notion of pattern systems for security and safety critical systems, as a first step for MBSA (Model Based Safety Analysis). We wish to promote a

[9] The approach is evaluated in the context of the TERESA project (http://www.teresa-project.org/)

framework to define reference models and patterns (sub-systems) for modeling and analysis of systems with strong security and safety requirements. The results will be provided in a *SEMCO* repository.

We also aim to build an experimental study in part of a software development environment based on UML. This is to judge the relevance of the artifacts produced for the assessment process.

ACKNOWLEDGMENT

This work is initiated in the context of *SEMCO* framework. It is supported by the European FP7 *TERESA* project and by the French FUI 7 *SIRSEC* project. In addition, we would like to thank the *TERESA* consortium who gave us valuable feedback on this paper.

REFERENCES

[1] A. Ziani, B. Hamid, and S. Trujillo, Towards a Unified Metamodel for Resources-Constrained Embedded Systems, in 37th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 485–492, IEEE, 2011.

[2] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, Security in embedded systems: Design challenges, ACM Trans. Embed. Comput. Syst., vol. 3, no. 3, pp. 461–491, 2004.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing, vol. 1, pp. 11–33, 2004.

[4] D. Schmidt, Model-driven engineering, in IEEE computer, vol. 39, no. 2, pp. 41–47, 2006.

[5] I. Crnkovic, M. R. V. Chaudron, and S. Larsson, Component- based development process and component lifecycle, in Proceedings of the International Conference on Software Engineering Advances (ICSEA 2006), p. 44, IEEE Computer Society, 2006.

[6] F. Buschmann, K. Henney, and D. Schmidt, Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing. Wiley, 2007.

[7] M. Schumacher, Security Engineering with Patterns - Origins, Theoretical Models, and New Applications, vol. 2754 of Lecture Notes in Computer Science. Springer, 2003.

[8] D. C. Schmidt and F. Buschmann, Patterns, frameworks, and middleware: Their synergistic relationships, in ICSE, pp. 694– 704, IEEE Computer Society, 2003.

[9] B. P. Douglass, Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Boston, MA, USA: Addison- Wesley Longman Publishing Co., Inc., 2002.

[10] B. Hamid, S.Gurgens, C. Jouvray, and N. Desnos, Enforcing S&D Pattern Design in RCES with Modeling and Formal Approaches, in ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS) (J. Whittle, ed.), vol. 6981, pp. 319–333, Springer, octobre 2011.

[11] B. Hamid, J. Geisel, A. Ziani, J. Bruel, and J. Perez, Model- Driven Engineering for Trusted Embedded Systems Based on Security and Dependability Patterns, in SDL Forum, pp. 72– 90, 2013.

[12] J. Gray, J.-P. Tolvanen, S. Kelly, A. Gokhale, S. Neema, and J. Sprinkle, Domain-Specific Modeling. Chapman & Hall/CRC, 2007.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework 2.0. Addison-Wesley Professional, 2nd ed., 2009.

[14] B. Hamid and C. Percebois, A modeling and formal approach for the precise specification of security patterns, in Engineering Secure Software and Systems - 6th International Symposium, ESSoS 2014, vol. 8364 of Lecture Notes in Computer Science, pp. 95–112, Springer, 2014.

[15] B. Hamid, A. Ziani, and J. Geisel, Towards Tool Support for Pattern-Based Secure and Dependable Systems Development, in ACadeMics Tooling with Eclipse (ACME), Montpellier, France, pp. 1–6, ACM DL, 2013.

Dr Brahim Hamid: *B. Hamid received his PhD degree from the University of Bordeaux (France), his PhD thesis was on the study of dependability in distributed computing systems. Then he worked at the CEA-Saclay List (French laboratory specialized on real-time embedded system modelling) where he worked on distributed component-based applications for dependable embedded and real-time. Currently, he is an associate professor at the University of Toulouse (France) and member of the IRIT-MACAO team. His main research topic is software languages engineering, at both the foundations and application level, for the development of secure and dependable distributed systems using model-driven engineering and design patterns. He participated in several research projects. In particular, he has led successfully the IRIT effort on the TERESA European project, and several national projects.*